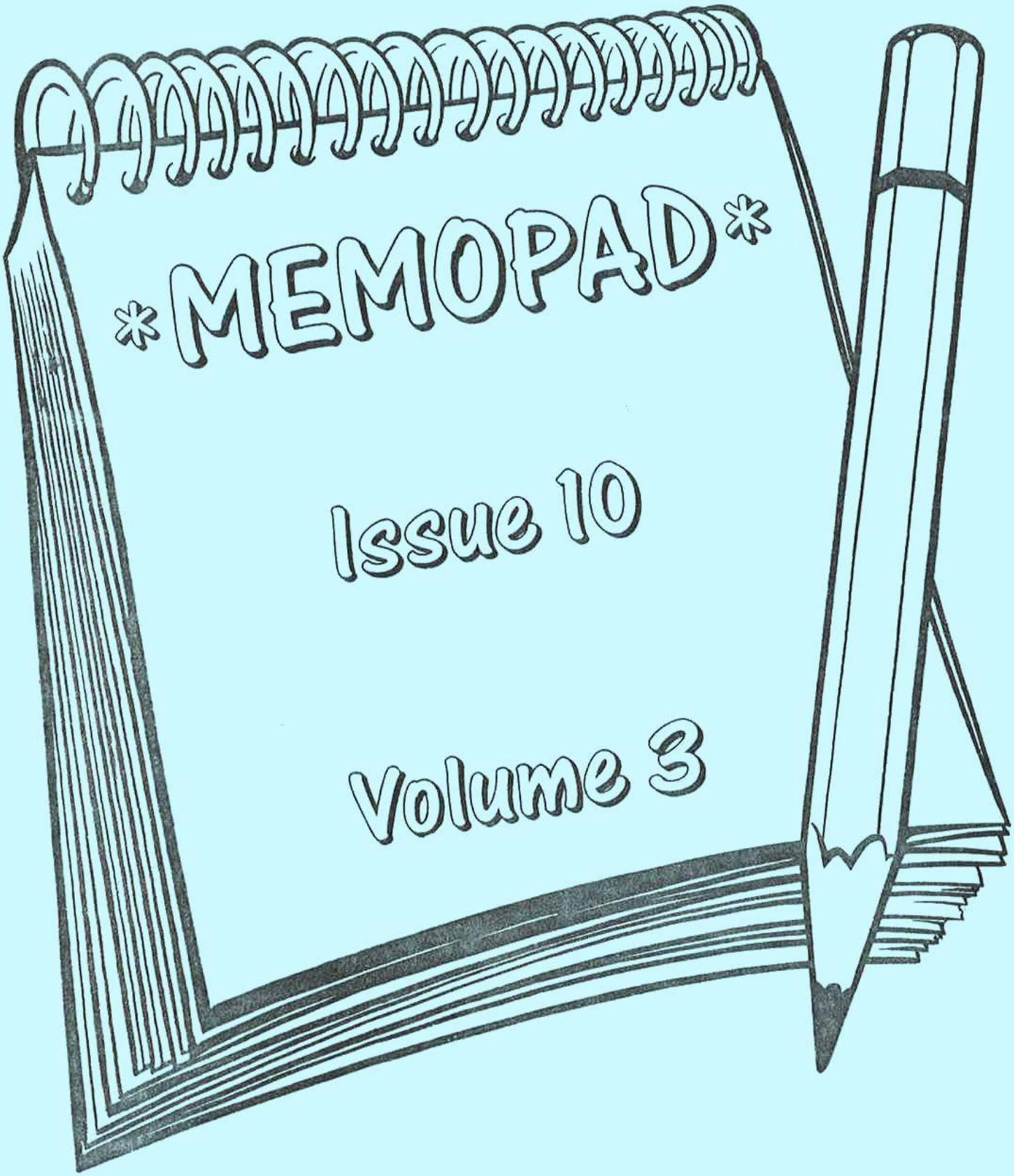

*MEMOPAD*
Issue 10
Volume 3

Volume Three **MEMOPAD** Number 10


Hello Readers,

Well! What can I say. The response from members has been overwhelming. I refer, or course , to how you have all responded to the new regime. Everyone who has rung the office has expressed there relief that someone has taken up the reins again. There have been, and will still be some problems but the faith you have expressed motivates us to strive for higher standards.

As well as the influx of new blood into the ranks, many of you are renewing your membership, although some are a little late I think this is largely due to the air of uncertainty which was very evident before Orion stepped in and took over the driving seat.

There is, I must admit, an internal conflict taking place at the moment. With the second print of The Source underway I have had to barricade myself in the print room so that I can get this edition of the magazine printed.

The atmosphere at Orion is one of happy optimism and everyone is working together to get things back on the right track. Gone are the days of nine to five - if the job wants doing it gets done, regardless of the time.

As you will all know the magazine is still not running on schedule, even though I seem to be churning editions out like strings of sausages. But it is looking a lot better and with a little luck and lot of hard work we will get there!

Sue

MEMOPAD IS PUBLISHED BY ORION SOFTWARE
FOR THE MEMOTECH USER GROUP

THE NORTHBRIDGE CENTRE
ELM STREET, BURNLEY,
LANCS. BB10 1PD
TELEPHONE - 0282 831695

COVER PRICE - £1.35 AVAILABLE BY SUBSCRIPTION ONLY



BARCLAYCARD AND ACCESS WELCOME

Number 10

# MEMOPAD

Volume Three

# *CONTENTS*

## What's In It For You!



NOTICE TO ALL MEMBERS

* RENEWAL OF SUBSCRIPTION *

PLEASE NOTE - A RENEWAL SLIP WILL BE ENCLOSED WITH THE MAGAZINE OF ANYONE WHO IS DUE TO RENEW THEIR SUBSCRIPTION. PLEASE COMPLETE THIS AND RETURN IT TO US AS SOON AS POSSIBLE SO THAT YOU DON'T MISS AN ISSUE OF THE MAGAZINE.

Volume Three **MEMOPAD** Number 10

# VIEW Letters TO THE EDITOR POINT

Dear Editor,

Your article written by Alan Wilson regarding Basic Tokens (issue 7 Vol. 3) is very misleading in the paragraph headed Multi-statement lines. In this paragraph, Alan Wilson infers that only the first command in a basic line is converted to a token and uses the example with the rem to demonstrate the same. Of course this is correct with a REM (on the same line).

However, try typing in the following three lines and then use the FRONT PANEL to examine the code as shown.

```
10  REM GOTO 10
20  VS 5: INK 4: PAPER 15: CLS
30  GOSUB 200: INK 7 : GOTO 10
```

The display will show the following codes (in HEX)

```
4000  0E 00 0A 00 80 20 47 4F
      54 4F 20 31 30 FF 10 00
      14 00 85 35 3A 8D 34 3A
      A4 31 35 3A 81 FF 10 00
      1E 00 97 32 30 30 3A 8D
      37 3A 96 31 30 FF
```

The code from 4000 to 400D is for line 10 (REM GOTO 10), and is as shown by Alan Wilson's article.

The code from 400E to 401D is for line 20 (VS 5 : INK 4: PAPER 15: CLS)The token 85 (HEX) or 133 decimal = VS, the token 8D (HEX) or 141 decimal = INK, the token A4 (HEX) or 164 decimal = PAPER and the token decimal 81 (HEX) or 129 decimal = CLS.

Similarly the code from 401E to 402D represents line 30 (GOSUB 200: INK 7: GOTO 10). The token 97 (HEX) or 151 decimal = GOSUB, and the token 96 (HEX) or 150 decimal = GOTO, the numbers being inserted in the basic line in ASCII (32 30 30 = 200).

Also a good point to remember is that the assembler uses a fair number of bytes in addition to the bytes used in a machine code programme, and to write GENPAT lines into machine code may not always save memory.

Yours faithfully

GORDON F. CARTER.

Dear Editor,

If I may, from an independent view, suggest something? I very much appreciate the recent main-stream conversions from Mastertronics, Adictive games, in my view this isn't really going to attract, say Spectrum, users over to the MTX. What would be more useful is to write new and original games on it which stand out from the average 8-bit game on other machines. Then with any luck a magazine ( most probably Popular Computing Weekly) will review it.

Even an average review would do some good. Remember, most mags. will not review a conversion, whose original has been covered before. Now that there are more machines about, it is probably less risky for Orion to invest in such a move.

I won't pretend to be a good programmer, but I am fanatical about the MTX - so much so that I sold my ST to get back to the 'Black Beauty', which I missed dearly. After all, why have a Porche when you are in love with a Metro - infinitely cheaper to run!

I am currently writing an adventure for the Series 2 which eventually I hope to release under my own label.

PHIL ARKLEY

Volume Three **MEMOPAD** Number 10

# The A.I Connection

Artificial Intelligence is supposed to be on the forefront of computer programming, yet to many it must seem about as relevant and as comprehensible as a Hex dump of machine code. This is unfortunate, because although at some levels it is almost ethereal, at a simpler level, it is eminently suitable for use in everyday programming, if there is such a thing.

As an example, consider Keith's recent 'Connect Four" program. Suppose we change it so that every time the game is played, the program tries to improve it's playing ability, although from the way I keep loosing that might not be easy. Anyway, we want it to improve, without having to change the way the moves are created by 'Computer Routine', and in a manner that might be applicable elsewhere. The best way to achieve this as I see it, is to use the G () array created in lines 1020 - 1030. These variables, as it says in the listing "affect the play'. Just how they affect the play isn't really important, so long as they do, and in a non-random manner. But how we can utilise the variables in an intelligent manner? and how do we check for improved playing.

What would be best, is if the playing ability improved gradually over time, and in a manner that is similar to a beginner who, at first doesn't play very well, but who hopefully, over time gets better and better by building upon their present level and play. So hopefully, the program should be able to build upon it's play by trying different ideas, and adopting only those that improve it's play. But at all times, the mayor part of it's should be based upon what it knows to work. In the case of "Connect Four", it's knowledge is contained in the G() array of 16 variables, whose values represent what the program currently knows about playing the game. As I said above, most of it's play should be based on it's current playing ability. So we could select one of these variables at random and change it's value, while leaving the rest as they are. The reason for choosing the variables at random rather than in sequence is that to change them in sequence would be a restriction that might prove detrimental to improving play. However, the new value for the variables could not be chosen at random, but be based upon the current value, so it should be changed by adding or subtracting a random value to it's current value. The range of values the variables can reach should be limited, so that the addition of the random value to the variables can reach should be limited, so that the addition of the random value to the variable near it's limit would not cause the program to crash. If we limit ourselves to + or - 1E30, this gives us a large safety margin. As for the random value and precision (There is a way round the problem and if our glamorous editor is willing (hint hint) maybe I could explain in a future edition).

Now the problem of checking for improved playing. The method described is about the simplest possible, and as such is limited in that it only really improves as its opponent improves, or is that an advantage? The limitation of this is that constant wins or constant looses have little lasting effect on the programs playing abilities. The method is this, if the program wins a game it is an improvement, while if it looses, it is not.

Anyway, that's the theory, what follows is the program changes necessary. It is a simple routine, and whilst being hardly on a par with 'C3PO' is is a first rung on the ladder.

```
40 GOTO 8000
5185 LET RESULT = -1: REM LOST
6192 LET RESULT = 0: REM DRAW
6305 LET RESULT = 1: REM WIN
7005 RETURN
```

PROGRAM ADDITIONS

```
8000 REM           CHANGE VARIABLE
8010 LET POS = INT (RND x 15) + 1
8020 LET MEM = G (POS) : REM STORE OLD VALUE
8030 LET VA = RND x 2000 : REM TWICE ADD ONN
8040 LET G (POS) = G (POS) + VA - 1000
8050 IF ABS (G (POS)) > 1 E30 THEN LET G (POS) = 1 E30 x SGN
     (G(POS))
8090 GOSUB 1500 : REM PLAY THE GAME.

8100 REM   CHECK RESULT
8110 IF RESULT > -1 THEN GOTO 8200
8115 REM    NO IMPROVEMENT
8120 LET G (POS) = MEM : REM RESTORE OLD VALUE

8200 REM REPLACEMENT FOR LINES 7010 TO 7040
8210 LET MES$ = "DO YOU WANT ANOTHER GAME?" : GOSUB 560
8220 IF IN$ = "N" THEN CLS : STOP
8230 IF IN$ <> "Y" THEN GOTO 8210
8240 REM  CAN'T USE CLEAR BECAUSE WE STILL NEED G().
8250 FOR I = 1 TO 8 : FOR J = 1 TO 8 : FOR K = 1 TO 2 :
     LET TG$ (I,J,K) = " "
8260 NEXT K : NEXT J
8270 LET R (I) = 0 : NEXT I
8280 FOR I = 1 TO 4 : LET A (I) = 0 : LET K (I) = 0 :
     LET J (I) = 0 : NEXT I
8290 LET X$ = " "
8300 VS 4 : COLOUR 2, 1 : CLS : COLOUR 4, 1 : COLOUR 0,
     1 : RAND -> 8
8310 GOSUB 500 : GOSUB 600 : GOTO 8000

8500 REM TO GET LONG TERM CHANGES, THE ARRAY G () WILL HAVE TO
8510 REM BE RESTORED AND SAVED AT THE START AND END OF PLAY.
```

In case you think anything so simple cannot be termed as Artificial Intelligence, A.I., as it is known, is defined as doing anything that if it were to be performed by humans, would require 'Intelligence'.

So how do you decide if you are performing better ?

# Is There Life On CP/M?

After reading the 'THAT'S LIFE' article in the latest MEMOPAD I wondered what it would look like on an 80-column screen, but didn't feel that finding out really justified doing it in 8080 code and converting it to CP/M I/O.Since a good modern Pascal compiler is about as fast as Forth there doesn't really seem a good reason for such effort,so I include the listing below.

The program allows you to choose between a random startup configuration and a DIY version.The latter asks for DEPTH and WIDTH parameters and you must then enter DEPTH strings each of maximum length WIDTH.Every character other than a space is treated as implying an occupied cell,and the program puts the resulting matrix in the middle of an empty screen to start the whole thing.

It was implemented on Hisoft's Pascal-80 compiler,but will run on any standard Pascal except for the random-number generator (use whatever is available : the idea is not to fill too many startup cells!).The rules for making a new generation are in the procedure 'transformcolony':if you want to alter them just change the case table.If you are using a full scale compiler you can even have several rule modules selected by conditional compilation.The options line should be selected for maximum run time speed.The listing is the actual source file and can be guaranteed to run.

```
{$L-,O-,I-,A-,S-,C-,T,Y                    }
PROGRAM gameoflife;
CONST
  nrows          = 18; nrows1     = 19;
  ncols          = 70; ncols1     = 71;
TYPE
  condition      = (unoccupied,occupied);
  rowrange       = 1..nrows; xrowrange    = 0..nrows1;
  colrange       = 1..ncols; xcolrange    = 0..ncols1;
  colonytype     = ARRAY [xrowrange,xcolrange] OF
                   condition;
VAR
  generation,ngenerations          : integer;
  colony                           : colonytype;
  state                            : (stillchanging,extinct,stable);

{ seed random generator fom keyboard                                    }

PROCEDURE randseed;
VAR
  c              : char;
  r,r1,r2        : integer;
BEGIN
  page;FOR r := 1 TO 11 DO writeln; r := 0;
  write ('ENTER ANY NUMBER FOR RANDOM SEED: ':40);
  read (r);
  r1 := r MOD 1001; r2 := r MOD r1;
  ranseed ( r,r1,r2 );
  page
END;
```

```
PROCEDURE setupcolony;
VAR
  row             : xrowrange;
  depth           : rowrange;
  col             : xcolrange;
  width           : colrange;
  ch              : char;
BEGIN
  write ('Random start? (Y/N) ');
  REPEAT
        readln;
        read(ch)
  UNTIL ch IN ['Y','y','N','n'];
  IF ch IN ['Y','y']
        THEN    BEGIN
        writeln;
        writeln('INITIALISING');
        FOR row := 1 TO nrows1 DO
                FOR col := 1 TO ncols1 DO
                        IF (random >= 0.9) THEN colony[row,col] :=
occupied
                                ELSE      colony[row,col]       :=
unoccupied;
        writeln;
        write('Generations: ');
        read (ngenerations);ngenerations := abs(ngenerations);
        generation := 1; state := stillchanging;
        page
        END
  ELSE  BEGIN
  write('Generations,Depth,Width: ');
  read(ngenerations,depth,width);ngenerations                    :=
abs(ngenerations);
  generation := 1; state := stillchanging;
  FOR row := 1 TO nrows1 DO
        FOR col := 1 TO ncols1 DO
                colony [row,col] := unoccupied;
 FOR row := (nrows - depth) DIV 2 + 1 TO (nrows + depth) DIV 2 DO
        BEGIN
        col := (ncols - width) DIV 2 + 1;
        readln;
        REPEAT
                read(ch);
                IF ch <> ' ' THEN
                        colony [row,col] := occupied;
                col := col + 1
        UNTIL eoln
        END
  END
END { OF SETUPCOLONY } ;

PROCEDURE drawcolony;
CONST
  line  = '-'; cross     = '+'; vline     = '!';
  organism      = '*'; nonorganism      = ' ';
VAR
  row                   : rowrange;
```

```
    col                     : colrange;
  BEGIN
    page;
    writeln;
    writeln( 'GENERATION ',generation:1);
    write( cross);
    FOR col := 1 TO ncols DO
          write(line);
    writeln(cross);
    FOR row := 1 TO nrows DO
          BEGIN
          write( vline);
          FOR col := 1 TO ncols DO
                IF colony [row,col] = occupied THEN
                      write(organism)
                ELSE write(nonorganism);
          writeln(vline)
          END;
    write( cross);
    FOR col := 1 TO ncols DO
          write(line);
    writeln(cross)
  END { OF DRAWCOLONY };

  PROCEDURE movetonextgeneration;
  VAR
    oldcolony      : colonytype;

    PROCEDURE transformcolony;
    TYPE
      zeroto8      =  0..8;
    VAR
      row          : rowrange;
      col          : colrange;

          FUNCTION neighbours:zeroto8;
          VAR
            n      : 0..9;
            dr     : xrowrange;
            dc     : xcolrange;
          BEGIN
          n := 0;
          FOR dr := row - 1 TO row + 1 DO
            FOR dc := col - 1 TO col + 1 DO
                  n := n + ord(oldcolony [dr,dc]);
          neighbours := n - ord(oldcolony [row,col])
          END;
    BEGIN { TRANSFORMCOLONY }
      FOR row := 1 TO nrows DO
          FOR col := 1 TO ncols DO
                CASE neighbours OF
                0,1,4,5,6,7,8 : colony [row,col] := unoccupied;
                2 :    ;
                3 : colony [row,col] := occupied
                END
    END { TRANSFORMCOLONY };
```

```
 PROCEDURE testforextinction;
 LABEL          1;
 VAR
   row          : rowrange;
   col          : colrange;
 BEGIN
  FOR row := 1 TO nrows DO
    FOR col := 1 TO ncols DO
       IF colony [row,col] = occupied THEN GOTO 1;
  state := extinct;
1 : END;

PROCEDURE testforstability;
LABEL  1 ;
VAR
  row           : rowrange;
  col           : colrange;
BEGIN
  FOR row := 1 TO nrows DO
    FOR col := 1 TO ncols DO
       IF colony [row,col] <> oldcolony [row,col] THEN GOTO 1;
  state := stable;
1 : END;

PROCEDURE wrap;
VAR
  r                       : xrowrange;
  c                       : xcolrange;
BEGIN
        FOR r := 0 TO nrows1 DO
                BEGIN
                colony[r,1] := colony[r,0];
                colony[r,0] := colony[r,ncols1];
                colony[r,ncols1] := colony[r,ncols]
                END;
        FOR c := 0 TO ncols1 DO
                BEGIN
                colony[1,c] := colony[0,c];
                colony[0,c] := colony[nrows1,c];
                colony[nrows1,c] := colony[nrows,c]
                END
END;

BEGIN
  oldcolony := colony;
  wrap;
  transformcolony;
  generation := generation + 1;
  testforextinction;
  IF state <> extinct THEN testforstability
END;


FUNCTION finished:boolean;
BEGIN
        finished := (generation = ngenerations) OR
                (state <> stillchanging)
```

# Saving Space

I had always wondered why so many published listings have pages of formatted print statements for a few hundred lines of code until some recent magazine letters showed that the well-known professional programmers' answer to this is not very well-known.

So here is the solution in two popular versions. The data for the 'listme' routines are ASCII files written with your favourite text-editor which need not be completed until you have written your program and can be painlessly edited as many times as you like.If your compiler supports the UNIX filename format you can even hide the textfiles from superficial prying by putting them in a different user area.In any case you can save many kilobytes of system memory and make your runtime descriptions as long as you like.

Both demo programs are actual source code : you must supply your own compiler's job-control instructions.There should be no problems with the 'C' version,but Pascal users should note:

1.some compilers will let you put parameter type definitions in the argument list and have local file variables,but you really shouldn't!

2.Pascal has no standard way of initialising a filename or assigning it to a logical file.The method below is common,but you should consult your manual.

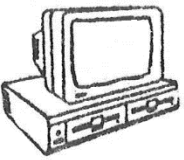
3.The addressed file must exist:Pascal has no standard way to detect or correct a file error.

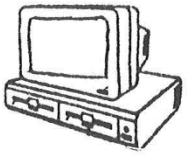
```
#include                    stdio.h

main()
{
        if (!listme("reversi.hlp"))
                {
                printf("\nCan't open input file");
                exit (0);
                }
}

int     listme(fname)
char    *fname;
{
        static int      count = 0;
        FILE    *fp;
        int     c;

        if ((fp = fopen(fname,"r")) == NULL)
                {
                return (0);
                }
```

```
  else
          {
          while ((c = getc(fp)) != EOF)
                  {
                  if (c == '\n') count++;
                  if (count != 0 && !(count % 23))
                          {
                          printf("\nRETURN");
                          getchar();
                          printf("\13\15\5");
                          count++;
                          };
                  putchar (c);
                  };
          };
  fclose(fp);
  return (count);
```

```
PROGRAM listme;
TYPE
        fnt                                     = ARRAY [1..14] OF char;
VAR
        infile                          : text;

PROCEDURE listme (f     :       fnt);
VAR
  c,ch                          : char;
  count           : integer;
BEGIN
  count := 0;
  reset (infile,f);
  WHILE NOT eof (infile) DO
        BEGIN
    c := infile^; get(infile);
        IF (c = chr(13)) THEN   BEGIN
                                writeln;
                                count := count + 1
                                END;
        IF ((count > 0) AND (count MOD 23 = 0))
        THEN    BEGIN
                        write('RETURN':40);
                        readln;
                        count := count + 1;
                        write(chr(11),chr(13),chr(5))
                        END;
        write(c)
        END
END;

BEGIN
        listme ('A:BIOMORPH.PAS');
END.
```

```
END;

PROCEDURE writeoutreason;
BEGIN
        IF state = extinct THEN
                writeln( 'THE COLONY HAS DIED.')
        ELSE IF state = stable THEN
                writeln( 'THE COLONY HAS BECOME STABLE.')
        ELSE
                writeln( 'THE SIMULATION HAS ENDED.')
END;

BEGIN
 randseed;
        REPEAT
                setupcolony;
                drawcolony;
                REPEAT
                        movetonextgeneration;
                        drawcolony
                UNTIL finished;
        writeoutreason
 UNTIL eof
END.
```





# TRACE

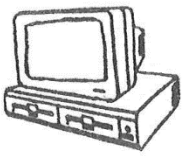
THIS MACHINE CODE ROUTINE WILL PRINT THE CURRENT BASIC LINE NUMBER WHEN THE 'BRK' KEY IS USED TO STOP A BASIC PROGRAM. IT IS PRINTED AT THE TOP R/H CORNER OF VS 1(THE LIST SCREEN) BETWEEN CHEVRONS, WHEN READY APPEARS.
IT IS NOT VERY ELEGANT - I KEPT IT AS SHORT AS POSSIBLE FOR EASE OF TYPING IN.
TO USE:-
LOAD YOUR (TROUBLESOME ?) BASIC PROGRAM AND ASSEMBLE THIS TO LINE ZERO (MUST BE AT START OF MEMORY!).
ENTER THE FOLLOWING POKES DIRECTLY :- POKE 64154,64(128-MTX512) POKE 64153,7
POKE 64152,195 . THIS WILL POINT A USER INTERUPT TO THIS ROUTINE.
POKE 64862,31 TO ENABLE TRACE. POKE 64862,15 TO STOP IT.

...PHIL AJKMAN....GENPAT 000632.....

```
4007            PUSH AF          ; SAVE MAIN REGS.
4008            PUSH BC
4009            PUSH HL
400A            PUSH DE
400B            LD A,£FE         ; SCAN KEYBOARD FOR BRK
400D            OUT (05),A
400F            IN A,(06)
4011            BIT 0,A          ; TEST IF SET
4013            JR NZ,EXIT       ; GOTO EXIT IF NO BRK
4015            LD HL,(£FD6A)    ; GET ADDR. OF CURRENT BASIC TOKEN
4018            DEC HL           ; SKIP BACK & LOOK FOR END OF PREVIOUS LINE
4019 LOOP:      DEC HL
401A            LD A,(HL)        ; FF IS END OF LINE
401B            CP £FF
401D            JP NZ,LOOP       ; KEEP LOOKING IF MULTI STATEMENT LINE
4020            INC HL
4021            INC HL           ; SKIP FWD TO LINE NO.
4022            INC HL
4023            LD C,(HL)        ; PUT LINE NO. IN BC
4024            INC HL
4025            LD B,(HL)
4026            CALL £0DDD       ; CONVERT BC TO ASCII CHAR. AT (DE)
4029            LD A,£EA         ; FIND NO. OF ASCII CHARS.
402B            SUB E
402C            LD B,A           ; PUT IN B
402D            LD A,(£FF58)     ; SAFE TO WRITE TO SCREEN?
4030            OR A
4031            JP NZ,EXIT       ; TRY LATER IF NOT
4034            LD A,32          ; SET SCREEN ADDR. TO CSR 0,32
4036            OUT (2),A
4038            LD A,£5C
403A            OUT (2),A
403C            LD A,"<"         ; PRINT <
403E            OUT (1),A
4040 LOOP1:     LD A,(DE)        ; PRINT ASCII CHARS.(LINE NO.)
4041            INC DE
4042            OUT (1),A
4044            DJNZ LOOP1       ; PRINT THEM ALL (SET IN B)
4046            LD A,">"         ; PRINT >
4048            OUT (1),A
404A EXIT:      POP DE
404B            POP HL           ; RESTORE REGS.
404C            POP BC
404D            POP AF
404E            RET
404F            RET

Symbols:
EXIT                    404A    LOOP    4019
LOOP1                   4040
```

....FOR THE MORE ADVENTUROUS THIS 'TRACE' CAN BE MERGED FAIRLY EASILY.
THIS IS WHAT YOU DO:- ASSEMBLE TRACE TO LINE ZERO WITH NO COMMENTS. SAVE IT.
WHEN NEEDED- LOAD TRACE FIRST. USE PANEL TO ADD £FB TO SYS VARIABLE 'VAZERO'.
THEN LOAD YOUR PROGRAM. NOW ADD £FB TO THE FOLLOWING SYS VARIABLES : 'NBTOP', 'BASTOP' AND 'BASTPO'. PUT ORIGINAL VALUE BACK INTO 'VAZERO'(IE: SUB. £FBF NOW)
THAT SHOULD HAVE MERGED THEM--- GOOD LUCK.

Volume Three **MEMOPAD** Number 10



# Utilising the SPK$ and DSI

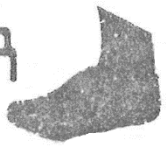
I have been looking at some uses for SPK$ and DSI commands and have developed this short programme which allows five pages of text to be typed and saved to tape. There are further options to send the pages to printer either singly in 40 or 80 columns, or all pages continuously in 40 or 80 columns.

To start the programme, simply load the tape which will auto run. The programme will dimension the computers memory to accept five pages of text. Start by selecting NEW PAGE and you will be presented with a blank screen, apart from the header, which gives the page number and an instruction to press <RET> to save the page in memory. You may now type in any desired text using all the control commands for DSI. It should be noted however, that PAGE MODE and CURSOR ON are set from within the programme and on no account should they be altered or parts of pages saved will be lost. If characters are accidentally typed in the header, do not worry because these are not saved with the rest of the text. Press <RET> to save the page, after which you will be returned to the menu. To start he next page, select EDIT PAGE and after making the necessary alterations, press <RET> to re-save the page. For a hard copy select SEND PAGES TO PRINTER and you will be presented with a second menu which gives you options for printing in 40 or 80 columns, single page or all pages continuously. To save your pages, select SAVE PROGRAM & PAGES, instructions will be given for saving to tape with an option to RETURN TO MENU. If you wish to re-use the programme and clear existing pages, simply break the programme at the MENU stage and type RUN.

```
1 REM**********************************
2 REM**** DSI TO PRINTER PROGRAMME ****
3 REM****      by David Bowles       ****
4 REM**********************************
5 DIM A$(5,22,39)
10 LET C=0: LET CC=0: LET Z=0: LET P=0: LET Y=0
20 VS 5: CLS : CSR 0,0: PRINT "LAST PAGE ENTERED,";CC;"  No
30 PRINT : PRINT : PRINT "  1.....NEW PAGE"
40 PRINT : PRINT "  2.....EDIT PAGES"
50 PRINT : PRINT "  3.....PRINT ON SCREEN"
60 PRINT : PRINT "  4.....SEND PAGES TO PRINTER"
80 PRINT : PRINT "  5.....SAVE PROGRAM & PAGES"
90 CSR 0,15: INPUT "  ENTER CHOICE  ";Z
100 IF Z=1 THEN  LET C=C+1: LET CC=C
110 IF C>5 THEN  LET C=C-1: LET CC=CC-1: PRINT "  NO MORE SPACE": GOTO 90
120 ON Z GOTO 20,200,400,400,600,2000,20,20,20,20
200 CLS
205 CSR 0,0: PRINT "PAGE No.";CC;"      PRESS <RET> TO SAVE PAGE": PRINT "------
----------------------------------------"
210 CSR 0,2: PRINT CHR$(27);"X";"^";: PRINT CHR$(27);"X";"J";: DSI
220 LET Y=1
230 CSR 0,Y+1
240 FOR N=1 TO 39
250 LET A$(CC,Y,N)=SPK$
260 NEXT
```

```
265 LET Y=Y+1
270 IF Y<23 THEN  GOTO 230
290 GOTO 20
400 PRINT : PRINT "NUMBER OF PAGES ON FILE...";C: PRINT
430 INPUT "ENTER PAGE NUMBER.........";CC
450 IF CC>C THEN  PRINT : PRINT "NO SUCH PAGE NUMBER": PAUSE 1000: GOTO 20
460 IF Z=4 AND P=1 THEN  GOTO 700
470 IF Z=4 AND P=2 THEN  GOTO 750
500 CLS : CSR 0,0: PRINT CHR$(27);"X";"J";: PRINT "PAGE No.";CC: PRINT "---------
--------------------------------"
510 FOR Y=1 TO 22: PRINT A$(CC,Y): NEXT
520 IF Z=2 THEN  GOTO 205 ELSE  GOTO 530
530 CSR 15,0: INPUT "PRESS <RET> FOR MENU";ANY$: GOTO 20
600 CLS : CSR 0,0: PRINT "YOU MAY PRINT ONE PAGE AT A TIME,": PRINT
610 PRINT "OR ALL PAGES CONTINUOUS."
620 PRINT : PRINT "  1.....SINGLE PAGE,40 COLUMNS"
622 PRINT : PRINT "  2.....SINGLE PAGE,80 COLUMNS"
624 PRINT : PRINT "  3.....ALL PAGES,40 COLUMNS"
628 PRINT : PRINT "  4.....ALL PAGES,80 COLUMNS"
630 PRINT : PRINT "  5.....RETURN TO MAIN MENU"
640 CSR 0,15: INPUT "ENTER CHOICE ";P
650 ON P GOTO 600,400,400,1000,1500,20,600,600,600,600
700 LPRINT CHR$(27);"D";CHR$(20);CHR$(0)
710 FOR Y=1 TO 22: LPRINT CHR$(9);A$(CC,Y): NEXT
720 GOTO 600
750 FOR Y=1 TO 22 STEP 2
760 LPRINT A$(CC,Y)+A$(CC,Y+1)
770 NEXT
780 GOTO 600
1000 LET CC=0
1005 LPRINT CHR$(27);"D";CHR$(20);CHR$(0)
1007 LET CC=CC+1
1010 FOR Y=1 TO 22: LPRINT CHR$(9);A$(CC,Y): NEXT
1015 IF CC<C THEN  GOTO 1007 ELSE  GOTO 20
1500 LET CC=0
1502 LET CC=CC+1
1505 FOR Y=1 TO 22 STEP 2
1510 LPRINT A$(CC,Y)+A$(CC,Y+1)
1520 NEXT
1525 IF CC<C THEN  GOTO 1502 ELSE  GOTO 20
2000 CLS : CSR 3,4: PRINT "TO SAVE PROGRAM & PAGES,"
2010 PRINT : PRINT "  SET TAPE MACHINE TO 'RECORD'"
2020 PRINT : PRINT "  AND PRESS <S>"
2030 PRINT : PRINT "  TO RETURN TO MENU, PRESS <M>"
2040 PAUSE 1000
2100 LET S$=INKEY$
2110 IF S$="" THEN  GOTO 2100
2120 IF S$="S" OR S$="s" THEN  GOTO 2130 ELSE  GOTO 20
2130 SAVE "DSI"
2140 GOTO 20
```

# FOOTBALL POOLS PREDICTOR



PART THREE

```
10 REM ********************************************************************************
20 REM ********************* FOOTBALL POOLS FORECASTING ***************************
30 REM ********************    PROFESSOR FRANK GEORGE    ***************************
40 REM ********************       COMMODORE 64           ***************************
50 REM ********************    ADAPTED FOR MEMOTECH      ***************************
60 REM ********************       BY DAVE WEMYSS         ***************************
70 REM ********************         JAN 1987             ***************************
80 REM ********************        POSITION.SUR          ***************************
90 REM ********************************************************************************
100 DISC SAVE "POSITION.SUR"
110 CLEAR : VS 5: CLS
120 DIM RECORD$(25,11,16),SAV$(12),DIV$(28),DATE$(12),LOA$(12),F$(2,12),TEMP$(11
,16)
130 LET SAV$="": LET DIV$="": LET DATE$="": LET F$(1)="F1.....": LET F$(2)=
"F2....."
140 CLS : PLOD "PROG1"
150 GOSUB 700: IF A<128 OR A>134 THEN GOTO 150
160 GOSUB 750
170 CLS : CSR 15,0: PRINT DIV$: GOSUB 850
180 DISC OPEN £1,LOA$,"1"
190 DISC INPUT £1,DATE$
200 CSR 70,0: PRINT DATE$: LET Y=0
210 GOSUB 1150
300 CSR 10,20: PRINT F$(1);"No sorting needed      ";F$(2);"Sort positions"
310 GOSUB 700: IF A<128 OR A>129 THEN GOTO 300
320 IF A=129 THEN GOTO 370
325 GOSUB 900
330 CLS : CSR 0,20: PRINT CHR$(5): CSR 10,20: PRINT F$(1);"Sort another division
     ";F$(2);"Return to main menu"
340 GOSUB 700: IF A<128 OR A>129 THEN GOTO 340
350 IF A=128 THEN GOTO 140
355 CLS : CSR 20,10: PRINT "Please wait.....": CSR 20,12: PRINT "Returning
to Main Menu"
360 DISC LOAD "POOLS.BAS"
370 CLS : CSR 15,0: PRINT DIV$: CSR 50,0: PRINT DATE$: PRINT
380 GOSUB 850
390 CSR 20,10: PRINT "Please wait.....": CSR 20,12: PRINT "Re-arranging le
ague positions"
400 FOR K=1 TO Y-1
410 FOR I=1 TO Y-1
420 FOR M=1 TO 11: LET TEMP$(M)="                    ": NEXT M
430 LET P=VAL(RECORD$(I,2)): LET Q=VAL(RECORD$(I+1,2))
440 IF P>=Q THEN GOTO 540
450 FOR M=1 TO 11
460 LET TEMP$(M)=RECORD$(I,M)
470 NEXT M
480 FOR M=1 TO 11
490 LET RECORD$(I,M)=RECORD$(I+1,M)
500 NEXT M
510 FOR M=1 TO 11
520 LET RECORD$(I+1,M)=TEMP$(M)
```

```
530 NEXT M
540 NEXT I
550 NEXT K
560 DISC OPEN £1,"TEAMS.SOR","O"
570 CLS : CSR 20,0: PRINT DIV$: CSR 50,0: PRINT DATE$: GOSUB 850: CSR 20,10: PRI
NT "Saving new league position "
575 DISC PRINT £1,DATE$
580 FOR I=1 TO Y
590 CSR 46,10: PRINT I
600 FOR X=1 TO 11
610 DISC PRINT £1,RECORD$(I,X)
620 NEXT X
630 NEXT I
640 DISC CLOSE £1

650 CSR 0,10: PRINT CHR$(5): CSR 20,10: PRINT "New league positions now saved"
660 DISC ERA LOA$
670 DISC REN LOA$="TEAMS.SOR"
680 CLS : GOSUB 900: GOTO 330
700 LET A$=INKEY$: LET A=ASC(A$)
710 RETURN
750 IF A=128 THEN LET LOA$="ENGDIVS1.TMS": LET DIV$="English First Division": LE
T N=22: GOTO 820
760 IF A=129 THEN LET LOA$="ENGDIVS2.TMS": LET DIV$="English Second Division": L
ET N=22: GOTO 820
770 IF A=130 THEN LET LOA$="ENGDIVS3.TMS": LET DIV$="English Third Division": LE
T N=24: GOTO 820
780 IF A=131 THEN LET LOA$="ENGDIVS4.TMS": LET DIV$="English Fourth Division": L
ET N=24: GOTO 820
790 IF A=132 THEN LET LOA$="SCOTPREM.TMS": LET DIV$="Scottish Premier Division":
 LET N=12: GOTO 820
800 IF A=133 THEN LET LOA$="SCOTDIV1.TMS": LET DIV$="Scottish First Division": L
ET N=12: GOTO 820
810 IF A=134 THEN LET LOA$="SCOTDIV2.TMS": LET DIV$="Scottish Second Division":
LET N=14
820 RETURN
850 FOR X=0 TO 79: PRINT "=";: NEXT X
860 RETURN
900 CLS : CSR 10,10: PRINT F$(1);"Make a printout      ";F$(2);"No printout"
910 GOSUB 700: IF A<128 OR A>129 THEN GOTO 910
920 IF A=129 THEN RETURN
930 CLS : CSR 30,0: PRINT DIV$: CSR 70,0: PRINT DATE$
940 GOSUB 850
950 CSR 20,18: PRINT "Check printer is on line!": CSR 20,20: INPUT "Press <P> to
 print > ";P$
960 IF P$<>"P" THEN GOTO 950
970 CSR 0,18: PRINT CHR$(5): CSR 0,20: PRINT CHR$(5)
980 LPRINT DIV$;"           ";DATE$
990 GOSUB 1100
1000 LPRINT CHR$(27);"D";CHR$(3);CHR$(20);CHR$(0)
1010 CSR 20,10: PRINT "Now printing position "
1020 FOR X=1 TO Y: CSR 42,10: PRINT X
1030 LPRINT RECORD$(X,1),RECORD$(X,2)
1040 NEXT X
1050 LPRINT CHR$(27);"@"
1060 RETURN
1100 FOR X=0 TO 79: LPRINT "-";: NEXT X
1110 RETURN
1150 LET S=N/2
```

```
 1160 FOR X=1 TO S
 1170 LET Y=Y+1
 1180 FOR I=1 TO 11
 1190 DISC INPUT £1,RECORD$(X,I)
 1200 NEXT I
 1210 CSR 9,X+4: PRINT X;".": CSR 14,X+4: PRINT RECORD$(X,1): CSR 31,X+4: PRINT R
ECORD$(X,2)
 1220 NEXT X
 1230 LET L=5
 1240 FOR X=S+1 TO N
 1250 LET Y=Y+1
 1260 FOR I=1 TO 11
 1270 DISC INPUT £1,RECORD$(X,I)
 1280 NEXT I
 1290 CSR 39,L: PRINT X;".": CSR 44,L: PRINT RECORD$(X,1): CSR 62,L: PRINT RECORD
$(X,2)
 1300 LET L=L+1: NEXT X
 1310 DISC CLOSE £1
 1320 RETURN
```

```
 PROG1
                          Division
* D MENU.                 ========

* R
English First Division.....................F1

English Second Division....................F2

English Third Division.....................F3

English Fourth Division....................F4

Scottish Premier Division..................F5

Scottish First Division....................F6

Scottish Second Division...................F7
```

## PART FOUR

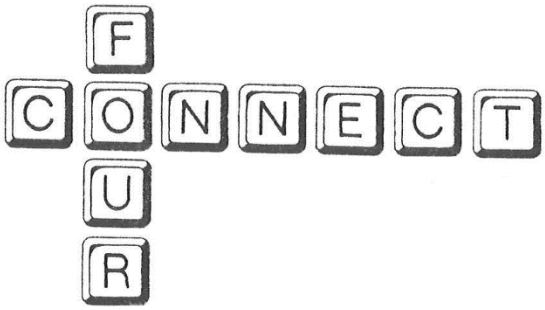
```
      ************************************************************************************
20 REM ********************* FOOTBALL POULS FORECASTING ***************************
30 REM *********************   PROFESSOR FRANK GEORGE   ***************************
40 REM *********************       COMMODORE 64         ***************************
50 REM *********************    ADAPTED FOR MEMOTECH    ***************************
60 REM *********************       BY DAVE WEMYSS       ***************************
70 REM *********************         JAN 1987           ***************************
80 REM *********************       INPUTTMS.BAS         ***************************
90 REM ************************************************************************************
100 DISC SAVE "INPUTTMS.BAS"
110 VS 5: CLS : CLEAR
120 DIM RECORD$(25,11,16),HEAD$(11,11),F$(2,12),B$(15),D$(2,20),DIV$(25),SAV$(12
),EMP$(15),D$(2,12),DATE$(9)
130 LET B$="": LET EMP$="                    ": LET F$(1)="F1.....": LET F$(2)="
F2.....": LET D$(1)="Another team": LET D$(2)="Finished"
140 RESTORE 850: FOR X=1 TO 11: READ HEAD$(X): NEXT X: GOSUB 900: GOSUB 1000
150 LET R=0: CLS : CSR 20,10: PRINT "Choose Division": PAUSE 2000: CLS
160 PLOD "PROG1"
```

```
170 GOSUB 400: IF A<128 OR A>134 THEN GOTO 170
180 GOSUB 510
190 DISC OPEN £1,SAV$,"O"
195 DISC PRINT £1,DATE$
200 CLS : CSR 20,0: PRINT DIV$: GOSUB 450
210 FOR I=1 TO 11: CSR 10,I+3: PRINT HEAD$(I): CSR 20,I+3: PRINT "> ": NEXT I
220 LET R=R+1: FOR I=1 TO 11: LET RECORD$(R,I)=EMP$: NEXT I
230 FOR I=1 TO 11: LET B$=""
240 CRVS 0,3,22,I+3,16,1,80
250 VS 0: EDITOR B$: IF B$="" OR LEN(B$)>15 THEN GOTO 250 ELSE LET RECORD$(R,I)=
B$
260 NEXT I: GOSUB 650
270 FOR I=1 TO 11
280 DISC PRINT £1,RECORD$(R,I)
290 NEXT I: VS 0: CLS
300 VS 5: CLS : CSR 10,20: PRINT F$(1);D$(1);"      ";F$(2);D$(2)
310 GOSUB 400: IF A<128 OR A>129 THEN GOTO 310
320 IF A=128 THEN GOTO 200
330 DISC CLOSE £1
340 CLS : CSR 20,10: PRINT F$(1);"Enter another division": CSR 20,12: PRINT F$(2
);"Return to Main Menu"
350 GOSUB 400: IF A<128 OR A>129 THEN GOTO 350
360 IF A=128 THEN GOTO 150
370 CLS : CSR 20,10: PRINT "Please wait.....": CSR 20,12: PRINT "Returning
 to Main Menu"
380 DISC LOAD "POOLS.BAS"
400 LET A$=INKEY$: LET A=ASC(A$): RETURN
450 FOR X=0 TO 79: PRINT "-";: NEXT X
460 RETURN
510 LET A=ABS(A-128)
520 ON A GOTO 530,540,550,560,570,580,590
530 LET SAV$="ENGDIVS1.TMS": LET DIV$="English First Division": GOTO 600
540 LET SAV$="ENGDIVS2.TMS": LET DIV$="English Second Division": GOTO 600
550 LET SAV$="ENGDIVS3.TMS": LET DIV$="English Third Division": GOTO 600
560 LET SAV$="ENGDIVS4.TMS": LET DIV$="English Fourth Division": GOTO 600
570 LET SAV$="SCOTPREM.TMS": LET DIV$="Scottish Premier Division": GOTO 600
580 LET SAV$="SCOTDIV1.TMS": LET DIV$="Scottish First Division": GOTO 600
590 LET SAV$="SCOTDIV2.TMS": LET DIV$="Scottish Second Division"
600 RETURN
650 VS 5: CLS : CSR 27,0: PRINT DIV$: GOSUB 450
660 FOR X=1 TO 11
670 CSR 10,X+3: PRINT HEAD$(X);":": CSR 22,X+3: PRINT RECORD$(R,X): NEXT X
680 CSR 20,20: PRINT "Any changes needed? (Y/N)": GOSUB 800
690 IF A$="N" OR A$="n" THEN RETURN
700 CSR 0,20: PRINT CHR$(5): CSR 20,20: INPUT "Line to change? (0 to change all)
 >";L: IF L<0 OR L>11 THEN GOTO 700
710 IF L=0 THEN GOTO 230

720 CLS : CSR 20,10: PRINT "Old Record: ";RECORD$(R,L)
730 LET B$=EMP$: LET RECORD$(R,L)=EMP$
740 CSR 20,12: INPUT "New Record (Max 15 chars) > ";B$: IF LEN(B$)>15 THEN GOTO
720 ELSE LET RECORD$(R,L)=B$
750 GOTO 650
800 LET A$=INKEY$: IF A$<>"" THEN GOTO 800
810 LET A$=INKEY$: IF A$="" THEN GOTO 810
820 IF A$<>"N" AND A$<>"n" AND A$<>"Y" AND A$<>"y" THEN GOTO 800
830 RETURN
```

```
850 DATA Team Name,Points,Played,HomeGoalsF,HomeGoalsA,AwayGoalsF,AwayGoalsA,Pla
yedH,PlayedA,LastMatch,Result
900 DISC OPEN £1,"HEADINGS.PLS","O"
910 FOR X=1 TO 11
920 DISC PRINT £1,HEAD$(X)
930 NEXT X
940 DISC CLOSE £1
950 RETURN
1000 CLS : CSR 20,10: INPUT "Date of entry of information > ";DATE$
1010 RETURN

PROG1

* D MENU.

* R      CHOOSE DIVISION MENU
         ====================

English First Division.............F1

English Second Division............F2

English Third Division.............F3

English Fourth Division............F4

Scottish Premier Division..........F5

Scottish First Division............F6

Scottish Second Division...........F7
```

TO BE CONTINUED..

---




By the time you have finished typing in the listing you will be able to view the results of your hard labours. However, don't be too enthusiastic to run the program - remember we have disabled interrupts so you will have to re-set so save it first!

When you run the program you should get a display with the CONNECT FOUR BOARD numbers and a heading *** CONNECT FOUR ***. If this appears, so far so good. If it doesn't go back and check all the code.

Before you start typing in the listing I must confess to finding a bug - no not really a bug it's a typing error. You will notice that I have started the listing with a little of the code from last month this is because I found, when testing it on two different machines that one machine corrupted so I have inserted the four PUSH HL, POP HL instructions. This slows the access time to VDP after sending out the address and also cures the problem.

Anyway, let's get the typing error out of the way. Find the PRINT subroutine which should be around £46E2. If you read this section you will see:

```
CP    £FF
JR    NZ,MESEND
```

Edit this line to read JR Z,MESEND or it will never ever print a string of characters! Silly me.

Now find the relevant code section to insert the PUSHES and POPS. It should be in the region of £474E depending on the comments you have used. The code reads :

```
IN    A,(01)
LD    E,8
```

At the line (£4750?) IN A,(01) go into the INSERT MODE and insert PUSH HL, PUSH hl, POP HL, POP HL. Exit from INSERT and go to the end of your file and start typing in the new code from SETBRD

When you have finished this listing find label START which should read.

```
DI
LD      SP,STACK
CALL    G2 INIT
CALL    CLS
```

After CALL CLS go into insert mode once more and type:

```
        CALL    SETBRD
TSTLP:  JR      TSTLP
```

The TSTLP is there so that when you run the program it will allow the display to stay on the screen and also stop the program crashing into the CLS routine.

Now don't be hasty! One more job. Find CHARX: where all the characters are stored. Now locate ;Z which is around £458C. Skip the next DB statement and then enter the EDIT MODE. Now start typing the Genpat statement from the BASIC LISTING in Issue 5. Don't forget to skip the first two values in the Genpat statement, they are only Genpat variables. Your last line of data from the Genpat statement should end immediately before:

```
        ; CHR$ (60H)  (96) dec
```

NOW SAVE IT! THEN TRY IT.

Next month we will start putting the guts of the program into the code and within a short time we shall have a program to play with.

```
Connect Four Assembler version for magazines only (c) K. Hook 1987     MACRO-80 3.44   09-Dec-81

                                TITLE Connect Four Assembler version for magazines only (c) K. Hook 1987
                                .LIST
 474F   E5                             PUSH    HL
 4750   E5                             PUSH    HL
 4751   E1                             POP     HL
 4752   E1                             POP     HL
 4753   DB 08                          IN      A,(8)
 4755   1E 08                          LD      E,8
 4757                           CALC2:

 4757   0F                             RRCA
 4758   1D                             DEC     E
 4759   15                             DEC     D
 475A   20 FB                          JR      NZ,CALC2
 475C   B7                             OR      A
 475D   C9                             RET                     ;RETURN WITH BIT PATTERN IN A


 475E                           SETBRD:
 475E   3E 81                          LD      A,81H           ;BLACK BACK GRND/RED FOREGROUND
 4760   32 471A                        LD      (PLTCOL),A      ;STORE IT
 4763   11 0010                        LD      DE,16           ;LINE 510 FOR I = 36 TO 164 STEP 16
 4766   0E 09                          LD      C,9             ;9 LINES TO PLOT

 4768   DD 21 0024                     LD      IX,36           ;INITIAL X POS
 476C   FD 21 0024                     LD      IY,36           ;INITIAL Y POS
 4770                           BRD1:
 4770   06 C0                          LD      B,192           ;192 POINTS  (36 TO 228)
```

```
4772  DD E5                     PUSH   IX            ;SAVE INITIAL X POS
4774                    BRD2:
4774  CD 46F0                   CALL   PLOT          ;PLOT A POINT
4777  DD 23                     INC    IX            ;NEXT POINT
4779  10 F9                     DJNZ   BRD2          ;DO TILL B = 0
477B  DD E1                     POP    IX            ;GET XPOS BACK
477D  0D                        DEC    C             ;DEC LINE COUNTER
477E  28 04                     JR     Z,BRD3        ;GO DO NEXT BIT
4780  FD 19                     ADD    IY,DE         ;OTHERWISE ADD HL TO GET NEXT Y POS
4782  18 EC                     JR     BRD1          ;AND DO THE WHOLE LOT AGAIN

4784                    BRD3:
4784  11 0018                   LD     DE,24         ;INCREMENT X WISE BETWEEN LINES
4787  0E 09                     LD     C,9           ;NUMBER OF LINES TO DRAW
4789  DD 21 0024                LD     IX,36         ;X POS STARTING POSITION
478D  FD 21 0016                LD     IY,22         ;Y POS   "       "
4791                    BRD4:
4791  06 8E                     LD     B,142         ;NUMBER OF VERTICAL PIXELS

4793  FD E5                     PUSH   IY            ;SAVE Y POS THIS TIME
4795                    BRD5:
4795  CD 46F0                   CALL   PLOT
4798  FD 23                     INC    IY
479A  10 F9                     DJNZ   BRD5

479C  FD E1                     POP    IY
479E  0D                        DEC    C
479F  28 04                     JR     Z,BRD6
47A1  DD 19                     ADD    IX,DE
47A3  18 EC                     JR     BRD4

47A5                    BRD6:                       ;LINE 36,22,228,22
47A5  06 C0                     LD     B,192
47A7  DD 21 0024                LD     IX,36
47AB  FD 21 0016                LD     IY,22
47AF                    BRD7:
47AF  CD 46F0                   CALL   PLOT
47B2  DD 23                     INC    IX
47B4  10 F9                     DJNZ   BRD7          ;ALL THE LINES ON BOARD NOW PLOTTED.

                        ;THIS SIMULATES LINE 530 OF THE BASIC LISTING BY PRINTING THE NUMBERS
                        ;IN THE SQUARES

47B6  DD 21 4297                LD     IX,XCORD      ;MAKE SURE IX =  CURSOR BUFFER IN KSUB1
47BA  3E 06                     LD     A,6           ;CONTROL CODE FOR CHANGE INK
47BC  CD 4199                   CALL KSUB1           ;SEND IT
47BF  3E 03                     LD     A,3           ;GREEN
47C1  CD 4199                   CALL   KSUB1         ;SEND  IT
47C4  DD 36 00 05               LD     (IX+00),5     ;CSR X POS 5
47C8  DD 36 01 14               LD     (IX+01),20    ;YPOS 20
47CC  0E 31                     LD     C,"1"         ;ASCII 1
47CE  06 08                     LD     B,8           ;1-8 TO PRINT
47D0                    BRD8:
47D0  79                        LD     A,C           ;CHARACTER IN A REGISTER FOR KSUB1
47D1  CD 4199                   CALL   KSUB1
47D4  DD 7E 00                  LD     A,(IX+00)     ;GET X POS
47D7  C6 02                     ADD    A,2           ;INCREMENT BY 2 BECAUSE KSUB1 INCS CSR POS
47D9  DD 77 00                  LD     (IX+00),A
47DC  0C                        INC     C            ;INCREMENT NUMBER
47DD  10 F1                     DJNZ   BRD8
```

```
                    ;PRINT CONNECT FOUR MESSAGE AT TOP OF SCREEN

47DF    DD 36 00 07         LD      (IX+00),7
47E3    DD 36 01 01         LD      (IX+01),1       ;LINE 540 CSR 9,1 BUT WE'VE ADDED SOME
                                                    ;CHARACTERS SO LESS 2 =7

47E7    DD 7E 04            LD      A,(IX+04)       ;GET CURRENT PAPER COLOUR
47EA    F5                  PUSH    AF              ;SAVE IT
47EB    DD 36 04 0F         LD      (IX+04),15      ;WHITE PAPER
47EF    21 413C             LD      HL,TITL         ;GET CONNECT FOUR MESSAGE
47F2    CD 46E2             CALL    PRINT           ;GO AND PRINT IT
47F5    F1                  POP     AF              ;GET PAPER COLOUR BACK
47F6    DD 77 04            LD      (IX+04),A       ;AND RESTORE IT
47F9    C9                  RET                     ;RETURN TO CALLER
```

---

# The Complete Price List

## Hardware

| DESCRIPTION | MEMBERS PRICE | NON MEMBERS PRICE | CARRIAGE |
|---|---|---|---|
| COMPLETE CP/M PACKAGE | | | |
| 1 X 1 MBYTE 3.5" INDUSTRY STANDARD DISC DRIVE, 500K FAST ACCESS RAM DISC CP/M 2.2 OPERATING SYSTEM. 256K RAM. 12" GREEN SCREEN MONITOR CENTRONICS STANDARD PRINTER I/F POSITIVE ACTION KEYBOARD. COLOUR MONITOR OUTPUT. TWO JOYSTICK I/F. | 359.95 | 399.95 | 20.00 |
| PURCHASES INDIVIDUALLY (basic system) | | | |
| 256K COMPUTER PLUS TAPE OPERATING SYSTEM | 89.95 | 99.95 | 10.00 |
| CP/M SYSTEM | | | |
| 1 X 1 MBYTE 3.5" DRIVE + 512 SILICON DISC + 80 COL + CP/M + N.W. | 237.59 | 264.00 | 10.00 |
| HX 12" GREEN SCREEN MONITOR | 85.49 | 95.00 | 10.00 |
| TWIN RS232 INTERFACE (UPGRADE) | 26.96 | 29.95 | 3.00 |
| FDX 2 X 1 MBYTE CP/M + 2 MBYTE SILICON DISC. | 877.50 | 975.00 | 10.00 |
| 32K MEMORY EXPANSION | 37.95 | 39.95 | 3.00 |
| 64K MEMORY EXPANSION | 47.45 | 49.95 | 3.00 |
| 128K MEMORY EXPANSION | 75.95 | 79.95 | 3.00 |
| NEWWORD ON ROM | 37.95 | 39.95 | 3.00 |
| PASCAL ON ROM | 37.95 | 39.95 | 3.00 |
| RS232 INTERFACE (FULL BOARD) | 37.95 | 39.95 | 3.00 |

SIDISC PRICES

| | | | |
|---|---|---|---|
| 1 X 1 MBYTE | 161.10 | 179.00 | 10.00 |
| 1 X 2 MBYTE | 304.20 | 338.00 | 10.00 |
| 1 X 3 MBYTE | 542.40 | 636.00 | 10.00 |
| PRINTER CABLE | 11.65 | 12.95 | 0.50 |

SPECIAL NOTES

Silicon Discs can be factory fitted for an extra £30.00 (U.K. Only).
FDX Twin Systems require RS232 Comms Board.
Carriage is applicable to U.K. orders only.
Always quote the type of computer owned Eg: MTX 500, 512 or series 2 when ordering add-on's.

** PLEASE NOTE ALL PRODUCTS WHICH ARE NOT MENTIONED ON THIS LIST ARE NO LONGER AVAILABLE **

THE ISSUE OF THIS PRICE LIST CANCELS ALL PREVIOUS OFFERS

| | | | |
|---|---|---|---|
| CRIB CARDS | £1.50 | ROM CALLS INFO SHEET | 0.50 |
| RST10 CALLS INFO SHEET | 0.50 | INTERRUPTS INFO SHEET | 0.50 |
| MTX SERVICE MANUAL | £9.95 | MTX NEW USER MANUAL | £8.95 |
| V.D.P. MANUAL | £7.95 | D.D.T. MANUAL | £2.50 |

DISC BASED LISTINGS

| | | | | | |
|---|---|---|---|---|---|
| MTX ROM LISTING | 5.25" | £15.95 | 3.5" | - | £21.95 |
| SDX DISC CONTROLLER | 5.25" | £9.95 | 3.5" | - | £12.95 |

ADVERTISING IN THE MEMOPAD

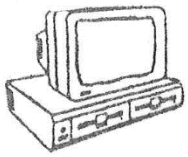
| | | | |
|---|---|---|---|
| SMALL ADVERT. | £5.00 | 1/8 PAGE | £27.50 |
| 1/4 PAGE | £45.00 | 1/2 PAGE | £80.00 |

THE COMPLETE HOME COMPUTER MAINTENANCE KIT

CASSETTE DEMAGNETISER
SCREEN + KEYBOARD CLEANER CLOTHS
CASSETTE HEAD CLEANER
DUST BLOWER
CLEANING WANDS

PACKAGED IN A PRESENTATION CASE
(MAKES A SUPER GIFT)

£21.50 + 50P P+P



Number 10

# MEMOPAD

Volume Three

# Software

U = Utility
G = Game
E = Educational
J = Joystick Compatible
L = Language
B = Business

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 3D TACHYON FIGHTER | G+J | ANY | 6.95 | MAXIMA | G+J | ANY | 6.95 |
| AGROVATOR | G | 512 | 5.95 | MEMOCHEQUE | U | 512 | 6.95 |
| ALICE | G | ANY | 6.95 | MEMOSKETCH | U+J | ANY | 7.95 |
| ASTROMILON | G+J | ANY | 6.95 | MEMOSKETCH SDX | U+J | 512 | 8.95 |
| ASSEM LANGUAGE COURSE | E | 512 | 8.95 | MISSION ALPHA. | G+J | ANY | 5.95 |
| ASTROPAC | G+J | ANY | 6.95 | MISSION OMEGA | G+J | ANY | 5.95 |
| KILLER TOMATOES | G | 512 | 7.95 | NEMO | G+J | ANY | 6.95 |
| BLOBBO | G+J | ANY | 6.95 | OBLIT. ZONE | G+J | ANY | 6.95 |
| BOUNCING BILL | G+J | ANY | 5.95 | OBLOIDS | G+J | ANY | 6.95 |
| BRIDGE | G | ANY | 6.95 | PAINTBOX | U | 512 | 5.95 |
| CAVES OF ORB | G | 512 | 5.95 | PAYROLL | B | 512 | 21.25 |
| CHAMBEROIDS | G+J | 512 | 6.95 | PHAID | G+J | ANY | 6.95 |
| CHESS | G | 512 | 8.95 | PONTOON | G | ANY | 6.95 |
| COMBAT | G | ANY | 3.95 | POT HOLE PETE | G+J | ANY | 6.95 |
| CRYSTAL | G | 512 | 6.95 | PURCHASE LEDGER | B | 512 | 12.75 |
| DISASM | U | 512 | 7.95 | QOGO | G+J | ANY | 6.95 |
| DOODLEBUG | G+J | ANY | 5.95 | QOGO 2 | G+J | 512 | 6.95 |
| DOWNSTREAM DANGER | G+J | 512 | 6.95 | QUANTUM | G+J | ANY | 5.95 |
| DR. FRANKIE | G+J | ANY | 5.95 | QUAZZIA | G+J | 512 | 6.95 |
| DRIVE THE CEE 5 | G+J | ANY | 6.95 | QUEST 1 | G | ANY | 5.20 |
| EDASM | U | 512 | 7.95 | REVERSI | G | 512 | 7.95 |
| EDASM SDX (DISC) | U | 512 | 8.95 | ROLLA BEARING | G+J | 512 | 6.95 |
| EMERALD ISLE | G | ANY | 6.95 | RUTHLESS B. | G | ANY | 3.45 |
| ESCAPE FROM ZARCOS | G+J | 512 | 6.95 | SALES LEDGER | B | 512 | 15.75 |
| EXTENDED BÅSIC | U | ANY | 7.45 | SALTY SAM | G+J | ANY | 5.95 |
| FATHOMS DEEP | G+J | 512 | 6.95 | SEPULCRI | G | 512 | 6.95 |
| FIG FORTH | L | 512 | 15.75 | SMG | G+J | 512 | 6.95 |
| FIG FORTH SDX (DISC) | L | 512 | 15.75 | SNAPPO | G+J | ANY | 6.95 |
| FIREHOUSE FREDDIE | G+J | 512 | 6.95 | SNOOKER | G | ANY | 7.95 |
| FIRST LETTERS 1 | E | 512 | 8.95 | SON OF PETE | G+J | ANY | 6.95 |
| FLUMMOX | G+J | 512 | 6.95 | SUPA CODER | U | 512 | 7.95 |
| GHOSTLY CASTLE | G | ANY | 3.45 | SUPER BIKE | G+J | ANY | 5.95 |
| GOLDMINE | G+J | ANY | 6.95 | SUPER MINEFIELD | G | ANY | 6.95 |
| GRAPHICS | U | ANY | 5.95 | SURFACE SCANNER | G+J | 512 | 6.95 |
| HEI-MATHS | E | 512 | 7.95 | TAPE TO DISC | U | 512 | 6.95 |
| HIGHWAY ENCOUNTER | G+J | 512 | 7.95 | TAPEWORM | G+J | ANY | 6.95 |
| HUNCHY | G+J | ANY | 5.95 | TARGET ZONE | G+J | 512 | 6.95 |
| ICEBERG | G+J | ANY | 5.95 | THE WALL | G+J | 512 | 5.95 |
| JUMPING JACK FLASH | G+J | 512 | 5.95 | TOADO | G+J | ANY | 6.95 |
| KARATE KING | G+J | ANY | 6.95 | TURBO | G+J | ANY | 6.95 |
| KILOPEDE | G+J | ANY | 6.95 | USER BASIC | U | 512 | 8.96 |
| KNUCKLES | G+J | ANY | 7.95 | USER BASIC (SDX) | U | 512 | 9.95 |
| LITTLE DEVILS | G+J | ANY | 5.95 | UTILITIES (SDX) | U | 512 | 9.95 |
| MISSILE COMMAND | G+J | ANY | 5.95 | VERNON & VAMPS. | G | ANY | 5.95 |
| MATHS 1 | E | 512 | 8.95 | WORD AND PICTURE | E | 512 | 8.95 |